**Problem 1**

The R program *dpareto.R* is as follows.

dpareto <- function(x, alpha, beta, log = FALSE) {

# Calculate the maximum length of inputs

L <- max(length(x), length(alpha), length(beta))

# Align all the inputs

x <- rep(x, length.out = L)

alpha <- rep(alpha, length.out = L)

beta <- rep(beta, length.out = L)

logdens <- ifelse(alpha <= 0 | beta <= 0,

NaN,

ifelse(x <= alpha,

log(0),

log(beta) + beta \* log(alpha) - (beta + 1) \* log(x)))

# Check whether NaNs exist

if (sum(is.nan(logdens)))

warning("NaNs produced")

if (log) logdens else exp(logdens)

}

I used the following code to test it.

dpareto(3, 2, 1)

dpareto(1, 2, 3)

dpareto(3, -2, 1)

dpareto(3, 2, -1)

dpareto(3 : 5, 2, 1)

dpareto(1 : 5, 2, 1)

dpareto(6, 2 : 4, 1)

dpareto(3, 2, 1, log = TRUE)

The results are as follows.

> dpareto(3, 2, 1)

[1] 0.2222222

> dpareto(1, 2, 3)

[1] 0

> dpareto(3, -2, 1)

[1] NaN

Warning message:

In dpareto(3, -2, 1) : NaNs produced

> dpareto(3, 2, -1)

[1] NaN

Warning message:

In dpareto(3, 2, -1) : NaNs produced

> dpareto(3 : 5, 2, 1)

[1] 0.2222222 0.1250000 0.0800000

> dpareto(1 : 5, 2, 1)

[1] 0.0000000 0.0000000 0.2222222 0.1250000 0.0800000

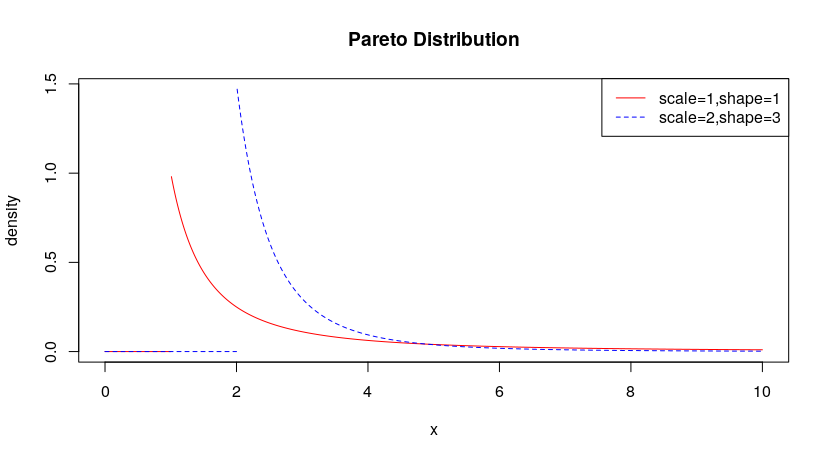
> dpareto(6, 2 : 4, 1)

[1] 0.05555556 0.08333333 0.11111111

> dpareto(3, 2, 1, log = TRUE)

[1] -1.504077

Then I used this function to plot the densities of tow sets of parameters in R.



**Problem 2**

The C program *paretodensDotC.c* and R program *paretodensDotC.R* are attached in Appendix. I used the following code to compile *paretodensDotC.c* and create shared objects for loading into R.

R CMD SHLIB paretodensDotC.c -Wall -pedantic

In Rstudio, I used dyn.load("paretodensDotC.so") to load the shared object file I just created. Then I can use the R function paretodensDotC I defined in *paretodensDotC.R* with .C interface to call my C function. I used the following code to test it.

paretodensDotC(3, 2, 1)

paretodensDotC(1, 2, 3)

paretodensDotC(3, -2, 1)

paretodensDotC(3, 2, -1)

paretodensDotC(3 : 5, 2, 1)

paretodensDotC(1 : 5, 2, 1)

paretodensDotC(6, 2 : 4, 1)

paretodensDotC(3, 2, 1, log = TRUE)

The results are as follows.

> paretodensDotC(3, 2, 1)

[1] 0.2222222

> paretodensDotC(1, 2, 3)

[1] 0

> paretodensDotC(3, -2, 1)

[1] NaN

Warning message:

In paretodensDotC(3, -2, 1) : NaNs produced

> paretodensDotC(3, 2, -1)

[1] NaN

Warning message:

In paretodensDotC(3, 2, -1) : NaNs produced

> paretodensDotC(3 : 5, 2, 1)

[1] 0.2222222 0.1250000 0.0800000

> paretodensDotC(1 : 5, 2, 1)

[1] 0.0000000 0.0000000 0.2222222 0.1250000 0.0800000

> paretodensDotC(6, 2 : 4, 1)

[1] 0.05555556 0.08333333 0.11111111

> paretodensDotC(3, 2, 1, log = TRUE)

[1] -1.504077

**Appendix**

***paretodensDotC.c***

#include <math.h>

#include <R.h>

#ifndef max

#define max( a, b ) ( ((a) > (b)) ? (a) : (b) )

#endif

/\*Define parto density function\*/

void paretodensDotC(double \*x, int \*nx, double \*alpha, int \*nalpha,

double \*beta, int \*nbeta, double \*dens, int \*lg)

{

/\*Calculate the maximum input length \*/

int i, n = max(max(nx[0], nalpha[0]), nbeta[0]), flag = 0, ind[3];

double ldens[n];

for (i = 0; i < n; i++) {

ind[0] = i % nx[0];

ind[1] = i % nalpha[0];

ind[2] = i % nbeta[0];

/\*Calculate whether alpha and beta are meaningful \*/

if (alpha[ind[1]] <= 0 || beta[ind[2]] <= 0) {

ldens[i] = NAN;

flag++;

}

/\*Check whether x is greater than alpha \*/

else if (x[ind[0]] <= alpha[ind[1]]) {

ldens[i] = log(0);

}

else {

ldens[i] =

log(beta[ind[2]]) + beta[ind[2]] \* log(alpha[ind[1]]) -

(beta[ind[2]] + 1) \* log(x[ind[0]]);

}

if (lg[0] == 0) {

dens[i] = exp(ldens[i]);

}

else {

dens[i] = ldens[i];

}

}

/\*Check whether NaNs exist \*/

if (flag > 0) {

warning("NaNs produced");

}

}

***paretodensDotC.R***

paretodensDotC <- function(x, alpha, beta, log = FALSE) {

nx <- length(x)

nalpha <- length(alpha)

nbeta <- length(beta)

n <- max(nx, nalpha, nbeta)

result <- .C("paretodensDotC",

as.double(x), as.integer(nx),

as.double(alpha), as.integer(nalpha),

as.double(beta), as.integer(nbeta),

dens = double(n), as.integer(log))

result$dens

}